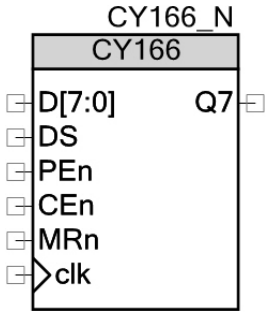
8-Bit Parallel-In/Serial-Out Shift Register

CY166 v1.0



Features

- Synchronous parallel load
- Synchronous clock enable
- Synchronous serial data input
- Asynchronous master reset
- Simple to deconstruct

General Description

This shift register is based on the 74HC166. It is an 8-bit shift register with a fully synchronous serial and parallel data entry selected by an active LOW enable (PEn) input. When PEn is LOW, parallel data is entered into the register. When PEn is HIGH, data is entered into the right side of the register from the serial data input (DS). The remaining bits are shifted. The clock has an enable (CEn) which allows the register to be held in a static or hold mode. MRn is an asynchronous active low master reset that overrides all other inputs.

This component was built as a teaching tool. This classic component's operation is well understood and this datasheet's function is to help understand how it was built through its deconstruction.

Pin Description

Pin	Type	Function
D[7:0]	inputs	parallel data input
DS	inputs	serial data input
PEn	Input	parallel load enable (active low)
CEn	input	clock enable input (active low)
MRn	input	asynchronous master reset (active low)
Clk	input	clock input
Q7	output	serial output from last stage

Function Table

Operating Modes	Inputs				Qn REG		Output
	MRn	PEn	CEn	clk	Q0	Q1-Q6	Q7
reset	0	x	x	x	0	0-0	0
parallel load	1	0	0	↑	D0	D1-D6	D7
serial shift	1	1	0	↑	DS	Q0-Q5	Q6
hold	1	x	1	↑	Q0	Q1-Q6	Q7

Deconstructing the Component

This component has connections to the Parallel Out (PO) and Parallel In (PI) interface. If you do not understand how they operate then first review the CY161 component. It also shows how to route a serial input into a register.

For this shifter there are four different operations.

- Reset
- Parallel load
- Serial load
- Hold (do nothing)

While the reset operation will be handled with datapath reset circuitry, the other three operations need to be implemented with datapath instructions.

Using PSoC Creator, open the CY166 example project to see the project schematic (*TopDesign.cysch*). It has a CY166 component connected to input switches, output LEDs, and a clock.

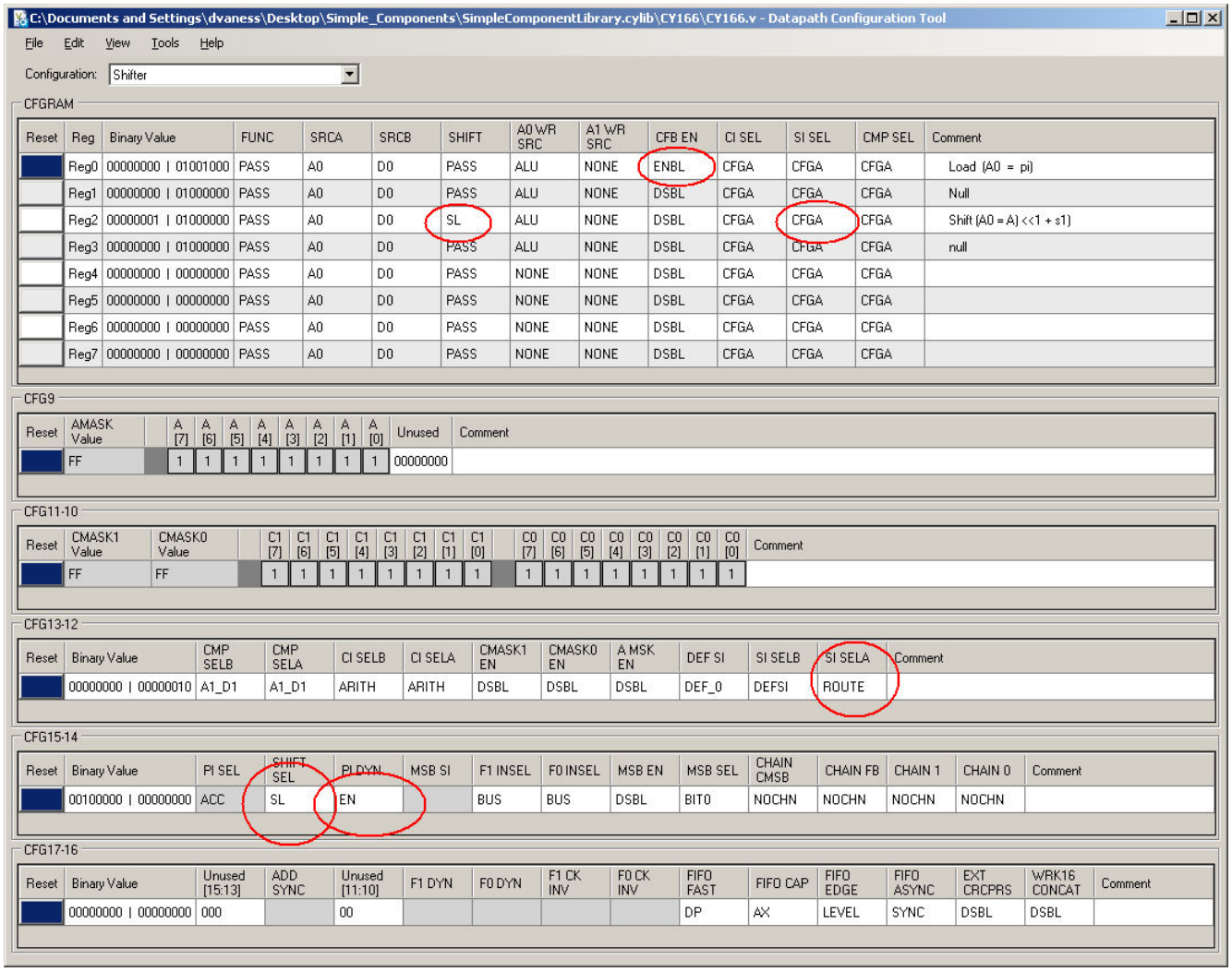
In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComps library project is, and select the CYCC_CY166_v1_0 component. Click **OK** and the following files are shown for the component:

- Symbol file (cysym)
- Datasheet (pdf)
- Verilog File (v)

Open the symbol file to find a symbol with six inputs and one output. It looks like the symbol shown on the first page. There are no additional symbol parameters.

Open the Verilog file and notice that at lines 28 – 34, these definitions were passed to this Verilog file when it was created. The first 22 lines of this header list register usage and the datapath instruction definitions.

There is a need for intermediate signals and this is handled in lines 38 - 39. What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



Note that although only three operations are needed to be implemented, four of the datapath instructions are used. By judicious selection of their address position and duplication, the logic to control them has been greatly simplified. The PI_DYN bit has been enabled so the CFB_EN bit of each instruction controls the ASRC input to the ALU. The datapath is configured for left shit operations (SO on left SI on right). SI SELA is set for ROUTE to connect the serial input to the shift register.

- For the load instructions the parallel input is passed through the ALU and placed in A0.
- For the null instructions A0 is passed through the ALU but not fed to any register. It does nothing.
- For the shift instruction A0 is incremented by the ALU and fed back to A0.

When saved, this tool inserts the updated configuration data back into the Verilog file along with an instance of the datapath interface. Just pass the correct signals in to and out of it and you are done.

This counter parameter list may look ominous, but only a few parameters have to be entered as shown below:

```
/* input         */ .reset(reset), // Connect reset signal to datapath
/* input         */ .clk(clk),     // Connect clock input to data path
/* input [02:00] */ .cs_addr({1'b0, PEn, CEn}), // control logic for instructions
/* input         */ .route_si(DS), // Connect serial data input to datapath
/* input [07:00] */ .pi(D[7:0]),   // connet input data to datapath PI
```

These parameters do the following:

- Connects a reset signal to the datapath reset circuitry
- Connects a clock signal to the datapath clock circuitry
- Control datapath instruction processing
- Routes the serial input signal into the shifter
- Connects the parallel in to the datapath P1

Support

PSoC Creator Community Components are developed and supported by the Cypress Developer Community. Go to www.cypress.com/CommunityComponents to discuss this and other Community Components.

© Cypress Semiconductor Corporation, 2013. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control, or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This work is licensed under a Creative Commons Attribution 3.0 Unported License.

http://creativecommons.org/licenses/by/3.0/deed.en_US

You are free to:

- Share — to copy, distribute and transmit the work
- Remix — to adapt the work
- Make commercial use of the work